

# **TCP/IP and radio amateurism**

## **A UBA-RST TCP/IP TaskForce project**

Gert Leunen, ON1BLU

Unie der Belgische Amateurzenders (UBA) – Radioclub Sint-Truiden; EURO<LINK

Leeuwerweg 34, B-3800 Sint-Truiden, Belgium

[on1blu@qsl.net](mailto:on1blu@qsl.net)

### **Abstract**

Although several reports on TCP/IP projects have already been published, we felt our specific approach and vision could still figure as a contribution to the subject. The approach we will present here is one that addresses nearly all aspects of the network infrastructure (from hardware solutions, through network topology, up to services), focuses on transparency to the user and provides INTERACTION with legacy – as opposed to PORTING legacy into TCP/IP.

### **Introduction**

Ever since Packet Radio saw its daylight during the eighties – with speeds up to 300 and 1200 baud's – it appears not to have evolved to better operation. Sure, there are many initiatives, 9600 baud G3RUH being the most persistent, but none of them is sufficiently wide spread. It's almost as if Packet Radio is the most noticeable sign of what's currently happening to radio amateurism in general (in Belgium at least): a fade out. Many discussions have been held on the causes, Internet and GSM being the most frequently referenced. In my opinion, there are 2 causes to this:

- Even though we have a wide spanning Packet Radio network (and the Packet Radio network currently is the only way to connect ALL HAMs on a 'PERMANENT' basis), the pace of its evolution now results in an arcane technology that's hard to work with. This has a negative effect on sharing technical knowledge and experience, and therefore prohibits large based experiments.
- Since radio amateurism hasn't evolved as the Internet did, for example, there's no need to experiment with new technologies. Transceivers are already very small, further improving them seems obsolete: such improvements won't let us talk faster :-), nor let us type faster (while working with our interactive BBS systems, to name just one).

The UBA-RST team concluded this particular Packet Radio domain should undergo a major evolutionary step. From the very beginning, we had two goals in mind:

- HAM-operators who are not interested in computers or Packet Radio itself - but want to use these TOOLS to exchange their experiences, should be able to do this in a very simple way, preferably using software they're already accustomed with (in other words: without having to learn various software packages).
- HAMs who feel to perform technical experiments should be provided a platform where they can learn things that are relevant to them. Let me illustrate: the Packet Radio protocol - AX.25 - is specifically designed for Amateur Radio. If technicians are forced to develop directly upon this AX.25 platform, they can hardly use their experience anywhere else. Ultimately, this means they must start producing their own goals, often not providing any contribution to radio amateurism in general.

So, now we come to the point why we choose for TCP/IP specifically:

- TCP/IP is the protocol that's currently being used on the Internet. If we superpose this protocol set on top of the bare AX.25 link protocol, we enable HAMs to use their favorite and accustomed Internet software (like Netscape Communicator, Microsoft Internet Explorer, Microsoft Outlook, etc). This would greatly simplify the usage of Packet Radio: not only by the related software being user-friendly itself, but also eliminating the need for learning additional software packages (the network and its services no longer dictate the client software to be used to exploit all its features – c.f. BBSs and terminal software supporting either YAPP or SPBIN protocols for binary transfers).
- TCP/IP is not only used on the Internet: even the smallest companies are switching to TCP/IP for use on their local networks (LAN). This means TCP/IP technology, TCP/IP knowledge and TCP/IP experience is as relevant as you can get! If you learned something on the AX.25 + TCP/IP network, you can apply this knowledge equally well in your professional life.

Especially the latter is very important, since it revives the "raison d'être" of radio amateurism. If one wants to perform real-life experiments on server technologies, he can only do this on the Amateur Radio network! Internet providers, companies and even universities WILL NOT allow you to experiment on their network (you could cause major damage when you suffocate their network and – as a consequence – potentially parts of the Internet as well). So, you could create an isolated network, but you can hardly call that a real-life test, can you?

### ***What the project is all about***

While we noticed many people were already experimenting with TCP/IP and high-speed Packet Radio, we never encountered a full-covering approach. So, even though our pure development efforts are somewhat limited (to the MCB-152 <sup>[1]</sup>), we generally collected many bits and pieces and tried to construct guidelines <sup>[2]</sup> for SysOps to bring their nodes back to life.

This was a step by step process as we'll illustrate shortly, but on realizing each individual step we considered one aspect thoroughly: LEGACY. None of our realizations would ever be accepted if they ruled out a group of users. So we must be compatible with existing Packet Radio systems (as far as modulation and coding is concerned), allow Packet Radio users to pass the system, allow mails and bulletins to be exchanged between TCP/IP users and BBS users, etc.

For the remainder of this text, we'll talk about AX.25 users and TCP/IP users. Of course, TCP/IP users also run AX.25, but they don't use the AX.25 protocol **consciously**.

Generally, Packet Radio was never meant to be the ultimate communication platform itself. If you take a look at the Digital Communications section of the well-known ARRL handbook <sup>[3]</sup> (even for an eighties-edition), you'll notice the OSI layered communication model (Table 1). While TCP/IP consists of lesser layers, we'll use the OSI layers bottom-up to tell our story step-by-step or layer-by-layer.

**Table 1 - The OSI reference model**

Application layer
Presentation layer
Session layer
Transport layer
Network layer
Data-link layer
Physical layer

### ***Raw Communication***

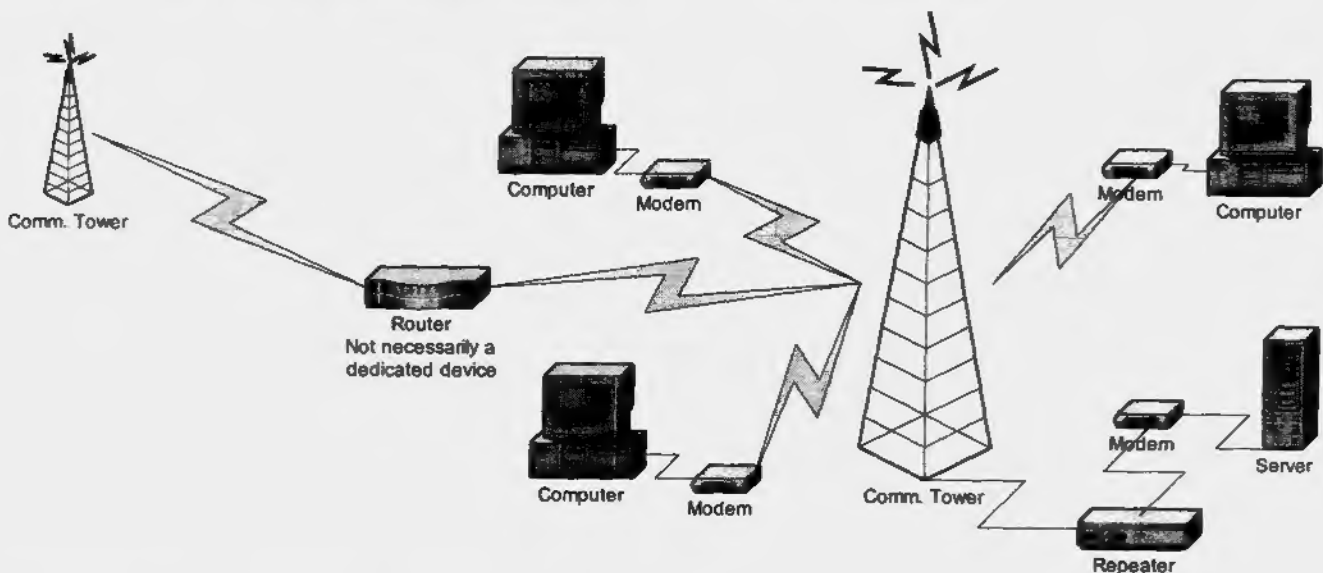
Let's start off with the physical layer. The physical layer is the grouping of communication media and the collection of equipment that's necessary for modulating and demodulating raw data on a specific

medium. For sound, the medium is the propagating vibration of molecules in a liquid or gas environment and human equipment are the vocal chords and ears. We, radio amateurs, are interested in the electromagnetic medium and the accompanying transceiver equipment, of course.

On our 9k6 user accesses at 70cm, the T7F transceiver <sup>[4]</sup> – yielding very good results – is typically used. For high speed user access, we're evaluating the 23cm data transceiver <sup>[5]</sup>, which should allow speeds up to nearly 200k baud.

For modulation and demodulation, we typically use BayCom <sup>[6]</sup> USCC-type modems (DK9RR and DF9IC modems). They can be plugged into our node's USCC card and onto our MCB-152 (as you'll see in the next section). The high speed user access tests are performed using BayCom's prototype high-speed DF9IC/DG3RBU modem.

We're experimenting with a repeater-like system (Figure 1), separating user-access uplink and downlink frequencies. This repeater-like system will perform either audio-repeating or echo-duplexing and possibly keep its transmitter on air (reducing TX/RX switching, which takes a considerable portion of the time required to send frames at these speeds). The major advantage of this system is that clients that are unreachable in simplex are now able to detect the other's signal, thereby significantly reducing the number of collisions (since 'invisibility' is a major cause for repeated collisions on user inputs). The node/router/server itself could use this repeater-like system as the users do, or could tap from the receiver of this "repeater" and send modulated data directly into its transmitter.



**Figure 1: Repeater setup for high speed with better collision avoidance**

If this repeater-like system succeeds, it could replace current expensive and dedicated links in some foreseeable future: if someone wants to start a new node, he only needs to buy one set of equipment (for linking up to the interregional repeater), instead of several sets for linking to several nodes. Interregional repeaters are then linked by nodes that volunteer linking to 2 interregional repeaters, essentially acting as routers at that time.

Generally, by adding an additional TCP/IP load to AX.25, we hope to create an essential need to further improve digital transmission performance.

## **Packet Radio**

At the data-link layer, we find our AX.25 protocol. The data-link layer is actually the collection of software that provides “courtesy” protocols (keep silent while others are talking, for example). When a TNC device is used, AX.25 is handled as part of that TNC’s firmware (next to command handling).

Our contribution to the link layer is the development of the MCB-152, which was mentioned earlier. These were our design goals when developing the MCB-152:

- it should support high speed communication evolution, without needing to buy entirely new equipment,
- for initial appreciation it must be compatible with current TNCs (1200 and 9k6 G3RUH-compatible) and current software (KISS firmware), and
- it should require minimal configuration effort.

The first two goals are realized by separating the controller hardware from the modem itself through a connection header that’s pin-compatible with BayCom’s USCC-type modems and by developing KISS firmware.

The third goal was realized by developing our famous SLIP firmware. After firmware download, the MCB-152 will act as an ordinary telephony modem (interpreting the highly standardized AT Hayes command set). The user dials as ‘telephone number’ the link address (call-sign) of his local IP router and as soon as the dial is complete, the MCB-152 will turn into a SLIP-to-AX.25 translator (and vice versa). So, the user only needs to setup an additional Internet account, using one of the generic modem drivers that are provided with Operating Systems like Microsoft Windows ‘95/’98/Me/NT/2000, etc. The MCB-152 also auto-detects the baud rate used on its serial port.

At the time of this writing, Joachim (ON1DDS) is re-developing the firmware for better performance, incorporating AX.25 V2.2 support.

One last word on our MCB-152: upon developing the firmware, we noticed the lack of supporting libraries for programming the Intel 80c152 chip. At that point, we decided not to limit the MCB-152 for TNC usage. Instead, Joachim (ON1DDS) developed a complete development library and collected all freely available development environments and documentation on a single CD. This enables the MCB-152 to be used for education, satellite tracking, you name it!

While the MCB-152 provides one solution to bridge the gap between popular Operating Systems and TNCs (by simulating commercial landline modems), SV2AGW <sup>[7]</sup>, FlexNet <sup>[8]</sup> and Linux adopt another solution: they provide drivers so the OS handles the TNC as just another LAN Network Interface Card. While we like the elegance of this solution, we noticed it’s generally hard to keep up with OS evolution: SV2AGW and FlexNet supports Windows ‘95/’98/Me(?), only recently Windows NT/2000 and probably not (yet) Windows XP. For Linux, I recently got the impression the networking part of the 2.4 kernel has been considerably revised and I still have to figure out its effect on my next-generation server. That’s why we thought there would be a reason for the existence of the MCB-152.

## **TCP/IP**

It’s only at the network layer that TCP/IP appears for the first time. You read this right: without AX.25, TCP/IP simply can not operate on our radio frequencies. The TCP/IP protocol set is available on nearly all computer architectures and operating systems.

For completeness:

- The network layer is the collection of software that handles routing. It's the network layer software that reduces our personal "network knowledge" to identifying our local IP router and identifying our destinations (no longer bothering about the local node of our destination and possibly several intermediate routes). The network layer software should also redirect data appropriately (and transparently!) when intermediate nodes fail.

When using the TCP/IP protocol set, IP (the Internet Protocol) is the routing mechanism used. It's a hop-to-hop routing protocol, meaning that routes are not calculated by the IP software on your machine. Instead, each node (including your machine) decides - for each individual packet - what neighboring node he will forward the packet to.

- The transport layer is responsible for transporting information. Until now, we only talked about packets of data; transport layer software will maintain an information stream on top of an unreliable network. Two kinds of transports are currently common: simple, unacknowledged datagrams (UDP protocol) and acknowledged in-order no-duplicate information transfer (TCP protocol).

In Belgium, we had to do some preparation, however. IP addresses were distributed at a too coarse level (provinces), and those provinces were assigned decimal coded identifiers. For as soon as a second TCP/IP server appeared in a single province, the problems started. Within a province, IP addresses were distributed sequentially. When several servers appeared, users typically operated through their nearest TCP/IP server. However, the IP address distribution didn't allow any binary mask to distinguish the users of different servers. Within a province, explicit host routes had to be created on all servers within that province, for all users in that province, resulting in large routing tables to be maintained manually.

Our TCP/IP TaskForce submitted a proposal <sup>[9]</sup> for introducing TCP/IP regions (corresponding to IP router availability). This allows for very simple network route entries. These regions are authoritative now, meaning that each region coordinator can distribute IP addresses freely within his assigned range of IP addresses. For now, region coordinators forward their assignments to the national IP coordinator, such that the latter can synchronize towards the worldwide [ampr.org](http://ampr.org) file.

In a near future, we hope **name server (NS) entries may be added to this worldwide [ampr.org](http://ampr.org) file**, thereby fully enabling the authority of the regions. Domain zones have already been introduced, however: my FQDN, for example, is [on1blu.baf.be.ampr.org](http://on1blu.baf.be.ampr.org) (even though there's a CNAME entry for [on1blu.ampr.org](http://on1blu.ampr.org) pointing to this 'regional' FQDN). The approach where all names are assigned to the 'root' [ampr.org](http://ampr.org) domain was sufficient in the (x)NOS days (which had no DNS 'system', only individual entries), but conflict with original DNS concepts as they are adopted in professional DNS servers (like BIND). Name server entries would also enable us to experiment with mechanisms such as DHCP/DDNS.

## **Sessions and presentation**

Currently, the TCP/IP layer model doesn't provide explicit session (some applications use cookies to simulate session management) and presentation layers (some applications implement MIME extensions), so we'll skip it for now.

## **Applications and services**

So, we finally reached the application layer. The application layer is the collection of all protocol API libraries (HTTP, FTP, SMTP, etc.), used by client and server software.

User applications (like Netscape Communicator, Microsoft Chat, etc.) use these API libraries for information exchange across the network. What this all means is that a programmer doesn't need to program a script engine for automating routes to distant BBSs, downloading and uploading mails and bulletins, or any other network issue for each application he wishes to create. He can now concentrate on one thing: providing a user-friendly interface.

These applications, however, communicate with service providers (Microsoft Outlook, for example, communicates with an SMTP server and a POP server). This brings us to our final effort: setting up TCP/IP servers.

We currently have 2 options:

- a Windows NT based server and
- a UNIX based server (like Linux)

The first isn't quite an alternative, especially in the NT4 era. NT4 is a desktop operating system with many networking tools. Next to the fact that NT servers are generally too expensive for amateur radio servers, many services need to be bought separately as well (all of them being equally expensive). Furthermore, a driver should be developed to let NT communicate with our TNCs, somewhat like SV2AGW and FlexNet32 do for Windows '95/'98, as SLIP can't be used on our servers (since it can only carry IP frames and no native Packet Radio traffic, breaking our support for legacy). A PPP firmware could solve this, however.

Windows '95/'98 is not feasible for providing services. Windows 2000, however, is a far better environment, since it really is a network operating system and it finally complies with standards. Most Internet services are included now too, but it's still expensive and requires driver development.

We choose for Linux since

- it's free,
- it supports AX.25 in its kernel NOW and
- it comes with all – currently well-known – services, all for free as well.

### ***The Linux server<sup>1</sup>***

Our project specifically collected configuration information for ALL those services. We succeeded in providing our users with a wealth of services:

- E-mail service, through the sendmail SMTP-server and the IMAP/POP3 server from the University of Washington
- News bulletin service, through the INN NNTP-server
- Web browser service through the Apache web (HTTP) server. We also allow our users to put their personal home pages on our web server
- FTP, NFS and SMB (c.f. Network neighborhood in Windows operating systems) file services, etc.
- Remote login (telnet) service
- Chat/conference service through Undernet IRC server

---

<sup>1</sup> Providing individual references that apply to this section would bring us to far. Please check the 'Related links' section on my website <sup>[2]</sup>.

- Domain name resolving through the BIND DNS server

Most of the server software mentioned is also very popular on the Internet for use by ISPs, universities, etc. Each of these services will facilitate information exchange enormously as schematics and source code can be mailed using drag & drop, project status can be published through web pages, etc.

Remember our permanent concern for legacy:

- We allow AX.25 users to participate in the IRC network.
- We also provide an Internet-type DX-cluster service that can be connected from both AX.25 and TCP/IP users (and which links up to the existing DX-cluster network): CLX.
- By using telnet, users are patched through to our node, so they can enter the legacy network (notice that you can't use legacy packet terminal software while you're dialed in on the same TNC, at least at the time of this writing for the MCB-152; PPP firmware and an appropriate 'TF TSR'-like hook would make this possible, however).
- And, extremely important, we provide an SMTP/NNTP <-> BBS gateway. Earlier, we ran a JNOS-box on our Linux server. This JNOS-box's TCP/IP stack was connected to the native Linux TCP/IP stack through a virtual SLIP link, allowing JNOS's SMTP and NNTP servers to exchange data with Linux's SMTP and NNTP servers. For forwarding, JNOS tapped Linux's AX.25 interface.

Now, we use the mailgw and lnxforward packages. Mailgw is actually an MTA (Mail Transport Agent) and we configure sendmail and INN to exchange BBS-destined or BBS-source messages with an FBB-compatible system.

Lnxfoward is an FBB-compatible forwarding-only module (also providing a web interface for reading mails and bulletins). This allows for transparent exchange of mails and bulletins between BBS and TCP/IP users.

Instead of using the lnxforward package, we could also install yet another full-fledged FBB system on our server, but we prefer not to. By installing another classical BBS system, you don't send a consistent message that users would benefit from switching to TCP/IP. Instead, we closed a deal with a neighboring BBS SysOp by which we have a forward feed to our gateway and our local AX.25 users have a home BBS through our local node step up. In Belgium, we also standardized the mapping between BBS areas and NNTP news groups (c.f. [9]). Altogether, this is a clear illustration of what we mean when we talk about "INTERACTING with legacy as opposed to PORTING legacy".

Once the configuration of our server was completed, a 5-day seminar was organized (attended by many Belgian SysOps), a step-by-step installation and configuration guide <sup>[2]</sup> was completed, and all related configuration files (as used at our server) were published on the web. The project currently refers to Red Hat Linux 5.2 based systems, but work is underway to update the project to more recent distributions.

## Conclusion

By introducing TCP/IP on our amateur frequencies, we brought radio amateurism back up-to-speed with industrial development/environment. As we noticed that major SQL (database) server builders like Sybase and Oracle make their latest versions freely available on Linux for development use and as we noticed free application servers – supporting Java and XML, which only recently exited hype stage in industry – for Linux, we are convinced radio amateurism can reconquer its pioneering role that characterized radio amateurism until a few decades ago.

To continue the illustration: for demonstrating TCP/IP at 76kbps, we issued voice-over-IP, which is still cutting-edge in industry (not that we have actual plans for applying this technology, but we did it!). And maybe this indicates the future of radio amateurism: at first, it was a technological hobby, then it partly became a communication hobby, but – mainly due to the Internet – it will return to its roots: technology through experiments.

## Reference

Members of the UBA-RST TCP/IP TaskForce:

- Gert Leunen (ON1BLU)  
*TCP/IP server setup*  
Belgian IP/DNS coordinator - Network 44.144.0.0 (0xffff0000) - Zone [be.ampr.org](http://be.ampr.org)  
AMPR-mail: [on1blu@on0baf.baf.be.ampr.org](mailto:on1blu@on0baf.baf.be.ampr.org)  
Internet E-mail: [on1blu@qsl.net](mailto:on1blu@qsl.net)
  - Joachim Elen (ON1DDS)  
*Firmware development*  
AMPR-mail: [on1dds@on0baf.baf.be.ampr.org](mailto:on1dds@on0baf.baf.be.ampr.org)  
Internet E-mail: [on1dds@qsl.net](mailto:on1dds@qsl.net)
  - Walter Machiels (ON4AWM)  
*Hardware development*  
AMPR-mail: [on4awm@on0baf.baf.be.ampr.org](mailto:on4awm@on0baf.baf.be.ampr.org)  
Internet E-mail: [walter.machiels@pandora.be](mailto:walter.machiels@pandora.be)
- [1] The MCB-152 project:  
Walter Machiels, ON4AWM (hardware: <http://home.worldonline.be/~vda10786>)  
Joachim Elen, ON1DDS (firmware: <http://www.8052.com/users/joachim.elen>)  
Distribution: <http://www.caseconsole.com/mcb152>  
Also read their paper "Revisiting the TNC firmware" in the proceedings of this conference edition
- [2] The Linux TCP/IP server configuration project:  
Gert Leunen, ON1BLU (<http://www.qsl.net/on1blu>, 'Linux server' section)  
Gert Leunen, *Wireless Linux TCP/IP server seminar*, June 2000 (downloadable from website)
- [3] The ARRL Handbook for Radio Amateurs, American Radio Relay League, yearly publication.
- [4] Holger Eckardt, DF2FQ, *A synthesizer-controlled 70cm-transceiver for 9600bd packet radio*  
[DF2FQ@amsat.org](mailto:DF2FQ@amsat.org)  
<http://www.rlx.lu/~rl/t7f/HANDBUCH.HTM> (German)  
[http://www.la3f.no/prosjekt/t7f/t7f\\_e.html](http://www.la3f.no/prosjekt/t7f/t7f_e.html) (English)
- [5] Bas de Jong, PE1JPD, *23 cm data-transceiver for high-speed packet*  
<http://www.dutch.nl/bdj>
- [6] BAYCOM, Bavarian Packet Radio Group  
<http://www.baycom.org>
- [7] George Rossopoulos, SV2AGW (<http://www.raag.org/sv2agw>)
- [8] Gunter Jost, DK7WJ, *FlexNet* (<http://www.afthd.tu-darmstadt.de/~flexnet>)
- [9] Gert Leunen, ON1BLU, *Organizational agreements for setting up a wireless TCP/IP network in Belgium (on top of AX.25)*, November 1999 (downloadable from website:  
<http://www.qsl.net/on1blu>, 'TCP/IP Taskforce' section, status track).